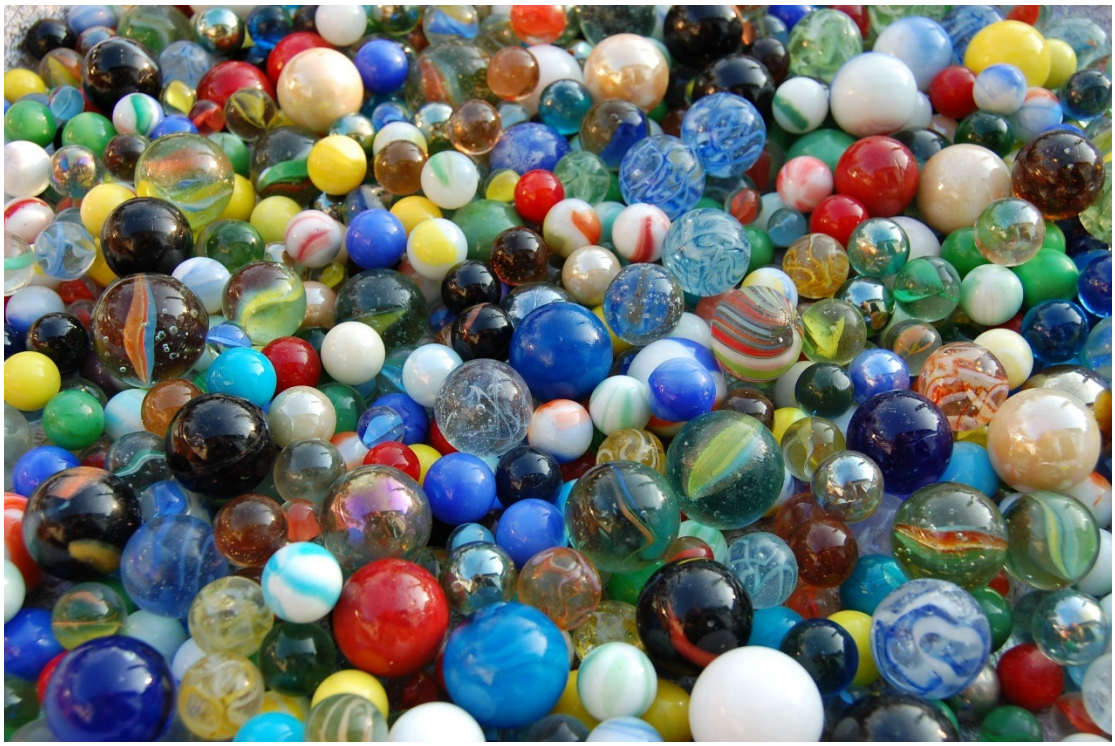
Assignment 2

Due Date

The assignment is due at 3PM Wednesday September 18th 2019

Background

Marbles! Large ones, medium-sized ones, small ones. Cats-eye ones, plain ones, swirly ones. Glass ones, wooden ones, plastic ones. Old ones, new ones. Marbles!



A Toy Company has asked you to develop an app to manage a club's collection of marbles, to analyse the data and illustrate various distributions. You will need to show the number of marbles per member, the number per size and style, *et cetera*.

There are seven members in the Club: Scott, Malcolm, Tony, Kevin, Julia, John, and Paul. Only marbles from these seven members will be stored. Marbles belonging to others should be ignored. All marbles belonging to the same member should be stored in the computerised collection together for ease of access.

Each marble should have its diameter in millimetres (a `double`), content (an enumeration), material (an enumeration), and age (a `bool`) stored. The collection of marbles for each member should be stored in increasing order of diameter (smallest first).

A text file exists with the basic data of the marbles found at the Clubhouse. I will give you code to read this into your program. You will need to illustrate (using a bar chart and/or some print outs) various information.

a Which underlying data structure (*array* or *linked-list*) will you use as a basis to model the collection of marbles for each of the seven members? In two–three sentences, justify your answer.

b Which kind of abstract data type (*binary tree, general tree, array, stack, priority queue, double-ended queue, set, list,* etc.) would you use to model the collection of marbles for each of the members? In two–three sentences, justify your answer by indicating the functionality required.

c Which underlying data structure (*array* or *linked-list*) will you use as a basis to model the collection of members? In two–three sentences, justify your answer.

d Which kind of abstract data type (*binary tree, general tree, array, stack, priority queue, double-ended queue, set, list,* etc.) would you use to model the collection of members? In two–three sentences, justify your answer by indicating the functionality required.

The types required to implement this include the following:

```
typedef struct {
     char *name;
     collection marbles;
} member;

typedef member members;

#define NUM_MEMBERS 7
const char *MEMBERS[] = { "Scott", "Malcolm", "Tony",
     "Kevin", "Julia", "John", "Paul" };

members club;
```

In other words, you must create a `struct` (called `member` in the above example) which combines the member name with your answer to (a) above to represent the collection of marbles for that particular member, and then use that as a basis for another type (used to create a variable called `club` in the above example) which uses your answer to (c) above to create a collection of `member`s.

A marble is defined as follows:

```
struct marble_int {
     double diameter;
     content look;
     material made_of;
     bool new;
};
```

```
typedef struct marble_int *marble;
```

and you may assume the existence of those types and the following types and functions:

```
typedef enum {plain, swirled, cats_eye} content;
typedef enum {glass, wooden, plastic} material;

void init_marble(marble *e, double d, content c,
    material m, bool n);
double get_diameter(marble e);
material get_look(marble e);
content get_material(marble e);
bool get_age(marble e);
void set_diameter(marble e, double d);
void set_look(marble e, content c);
void set_material(marble e, material m);
void set_age(marble e, bool n);
char *to_string(marble e);
```

A Visual Studio project is available on MyLO for you to download and use as a starting point. This comprises the following files:

- `marble.h` and `marble.c` — the *Marble* ADT as specified above. *These files are complete*;
- `assig_two219.c` — the file which contains the `main()` function and other functions which implement the required task (including reading marbles from the text file).

e You must complete `assig_two219.c`

Start by adding the *`collection`* and *`members`* types from above. You may add other types, constants and variables as necessary.

Then, complete the following functions:

- `get_member()` — which takes a member name, locates it in the `MEMBERS` array and returns its index (or –1 if it is not present);
- `add_existing()` — which takes the address of a non-empty collection of marbles for a particular member and an marble to add to that member's collection and which adds the marble to the collection in increasing order of diameter;
- `add_marble()` — which takes the name of a member and a marble and which either stores the name and marble in `club` if there are none for that member yet, or which calls `add_existing()` to do so if there are already marbles for that member;
- `process_category()` — which takes a collection of marbles for a particular member, a letter indicating which category should be processed, and an '`int`' indicating which value in the category the processing should occur for. Possible letters, values and actions are:

    - `'t'` — calculate the total number of marbles in the collection;
    - `'m'` — calculate the number of marbles of that particular material (plastic, wooden, or glass);
    - `'c'` — calculate the number of marbles of that particular content value (plain, swirled, or cats-eye); and
    - `'a'` — calculate the number of marbles of that particular age value (old or new).

  The answer (an `int`) should be returned.
- `show_graph()` — which takes a letter (see above), a string description of the output to be displayed (see sample output), and a value for the category (see above) or `-1` if the category is total — and which displays the histogram of, and totals (which can be obtained using `process_category()` for each member); and
- `main()` — which should initialize `club`, call `read_in_data()`, and call `show_graph()`.

If your answer to (a) or (c) above is a linked-list then you will also need to write `node.h` and `node.c` and add them to the project.

The project also contains the data file. This is just a text file which can be opened and read with most applications. It contains details of marbles found in the Clubhouse, but sometimes there are marbles which were left there by previous members.

Program specification

First you must obtain the marbles from a text file. The data must be stored in appropriate collections. At the top level there should be 7 members and for each there should be a collection of marbles (stored in ascending order of diameter).

As a marble is read from the file, the details should be checked for validity. If invalid — i.e. not belonging to a current member — it should be counted, but not included in the collection. If valid, its data should be stored in its members part of the collection.

Once the data have been read in and stored in the collections, output should be provided as specified in the preceding section.

A histogram illustrating the results per member should be displayed with one asterisk printed for every 25 marbles (other than for diameter — see sample output below).

Each histogram should state the count of ‘invalid’ marbles beneath it.

The output of your program should look something like the following (with the bold added in this document for emphasis only):

Marbles -- total number of

Invalid marbles: 1887

And the most common member for total number of marbles is......Paul!

Marbles -- glass

Invalid marbles: 1887

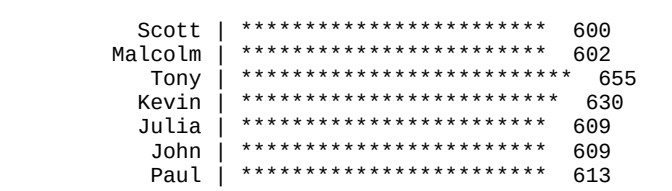
And the most common member for glass marbles is......Paul!

Marbles -- wooden

Invalid marbles: 1887

And the most common member for wooden marbles is......Paul!

Marbles -- plastic



Invalid marbles: 1887

And the most common member for plastic marbles is......Tony!

Marbles -- solid colour

Invalid marbles: 1887

And the most common member for solid colour marbles is......Paul!

```
                                    Marbles -- swirled colour

          Scott | **********************  575
        Malcolm | ************************  619
           Tony | ************************  614
          Kevin | ************************  617
          Julia | *************************  642
           John | ************************  620
           Paul | **********************  570

Invalid marbles: 1887


And the most common member for swirled colour marbles is......Julia!

                                    Marbles -- cats_eye pattern

          Scott | *************************  649
        Malcolm | ************************  611
           Tony | **************************  674
          Kevin | ************************  619
          Julia | *************************  644
           John | *************************  639
           Paul | ***************************  708

Invalid marbles: 1887


And the most common member for cats_eye pattern marbles is......Paul!

                                    Marbles -- old

          Scott | **********************************  911
        Malcolm | ************************************  973
           Tony | **********************************  944
          Kevin | ************************************  950
          Julia | **********************************  914
           John | **********************************  930
           Paul | ************************************  1010

Invalid marbles: 1887


And the most common member for old marbles is......Paul!

                                    Marbles -- new

          Scott | **********************************  911
        Malcolm | *********************************  885
           Tony | ************************************  967
          Kevin | **********************************  917
          Julia | ************************************  953
           John | **********************************  916
           Paul | **********************************  932

Invalid marbles: 1887


And the most common member for new marbles is......Tony!
```

Program Style

The program you write for this assignment must be contained in five files as follows:

- `marble.h` and `marble.c` for managing a marble;
- `node.h` and `node.c` for nodes of linked lists (if appropriate);
- `assig_two219.c` for the main algorithm which controls the reading of the data, the management of the collections, the calculation of the results, and the

display of the histogram. No function should be longer than about half a screen-full or so of code.

Your program should follow the following coding conventions:

- `const` variable identifiers should be used as much as possible, should be written all in upper case and should be declared before all other variables;
- `#define`d symbols should only be used for constant values if `const` is inappropriate, for example the size of an array.
- global variables should be used sparingly with parameter lists used to pass information in and out of functions.
- local variables should only be defined at the beginning of functions and their identifiers should start with a lower case letter;
- every `if` and `if-else` statement should have a block of code (i.e. collections of lines surrounded by `{` and `}`) for both the `if` part and the `else` part (if used)
- every `do`, `for`, and `while` loop should have a block of code (i.e. `{}`s)
- the keyword `continue` should not be used;
- the keyword `break` should only be used as part of a `switch` statement;
- opening and closing braces of a block should be aligned;
- all code within a block should be aligned and indented 1 tab stop (or 4 spaces) from the braces marking this block;
- commenting:
    - There should be a block of header comment which includes at least
        - file name
        - student name
        - student identity number
        - a statement of the purpose of the program
        - date
    - Each variable declaration should be commented
    - There should be a comment identifying groups of statements that do various parts of the task
    - Comments should describe the strategy of the code and should not simply translate the C into English

What and how to submit

What to submit

You must make *both* a paper and an electronic submission.

Paper submission

- A paper cover page (blanks can be collected from the Information and Communications Technology Discipline Help Desk and downloaded from MyLO).
- Written answers to parts (a)–(d) above.
- A *landscape oriented* print-out of `assign_two219.c` and any other program files you have added to the solution. *Your assignment will not be fully marked unless these are present.*

Electronic submission

- You should submit the entire Visual Studio project folder compressed as a ZIP file.

How to submit

Paper submission

- Firmly staple together all of the required documents (with the signed cover page on top) and place them in the appropriate submissions box near the ICT Help Desk.

Electronic submission

- You should ZIP the Visual Studio project folder and then submit it to the "Assignment 2" assignment drop-box on *KIT107*'s MyLO site.

Marking scheme

Task/Topic	Maximum mark
Design	
ADTs chosen wisely	3
Data structures correct and justified	3
Program operates as specified	
`node.c` correctly completed (if required, or array equivalent)	6
`main()` function shows abstraction of algorithm	2
Member index successfully found (`get_member()`)	2
Marble added —	
• As the first for each member (`add_marble()`)	2
• For an existing member (`add_existing()`)	2
• In increasing order by diameter (`add_existing()`)	3
• Discarding but counting invalid marbles (`add_marble()`)	1
Totals correctly calculated for each of six categories (`process_category()`)	4
Histogram displayed (`show_graph()`) —	
• With title	1
• With member name	1
• With asterisks (when appropriate)	2
• With total	1
• Number of discarded marbles displayed	1
• With 'winning' details	2
Program Style	
Does not unnecessarily repeat tests or have other redundant/confusing code	4
Uses correctly the C naming conventions	4
Alignment of code and use of white space makes code readable	4
Always uses blocks in branch and loop constructs	4
Meaningful identifiers	4
Variables defined at the start of functions only	4
Header comments for the program (author, date etc)	4
Each variable declaration is commented	4
Comments within the code indicate the purpose of sections of code (but DO NOT just duplicate what the code says)	4

Plagiarism and Cheating:

Practical assignments are used in the ICT discipline for students to both reinforce and demonstrate their understanding of material which has been presented in class. They have a role both for assessment and for learning. It is a requirement that work you hand in for assessment is substantially your own.

Working with others

One effective way to grasp principles and concepts is to discuss the issues with your peers and/or friends. You are encouraged to do this. We also encourage you to discuss aspects of practical assignments with others. However, once you have clarified the principles, you must express them in writing or electronically entirely by yourself. In other words you must develop the algorithm to solve the problem and write the program which implements this algorithm alone and with the help of no one else (other than staff). You can discuss the problem with other students, but not how to solve it.

Cheating

- Cheating occurs if you claim work as your own when it is substantially the work of someone else.
- Cheating is an offence under the Ordinance of Student Discipline within the University. Furthermore, the ICT profession has ethical standards in which cheating has no place.
- Cheating involves two or more parties.
    - If you allow written work, computer listings, or electronic version of your code to be borrowed or copied by another student you are an equal partner in the act of cheating.
    - You should be careful to ensure that your work is not left in a situation where it may be stolen by others.
- Where there is a reasonable cause to believe that a case of cheating has occurred, this will be brought to the attention of the unit lecturer. If the lecturer considers that there is evidence of cheating, then no marks will be given to any of the students involved. The case will be referred to the Head of the Discipline of ICT for consideration of further action.

Julian Dermoudy, August 27th 2019.

Version 1: 19/8/2019
Version 2: typographical corrections